ORACLE
FUSION MIDDLEWARE

An Oracle White Paper
April 3rd, 2013

Building Large-Scale eCommerce Platforms With Oracle

ORACLE

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Any statements in this document about current support are accurate as of release date of this document and may not be accurate at any time after that date. For currently supported environments, see the official support matrix.

Executive Summary

Building the platform under the largest eCommerce websites in the world requires special up-front design and robust scalable technology. Large-scale eCommerce websites today require hundreds or even thousands of CPU cores, which presents unique challenges. As a provider of technology to 94[1] of the top 100 e-Retailers in the U.S., Oracle supplies the technology and organizational expertise required to support the largest eCommerce websites in the world.

Oracle's solution for large-scale eCommerce websites (inclusive of the eCommerce application) consists of:

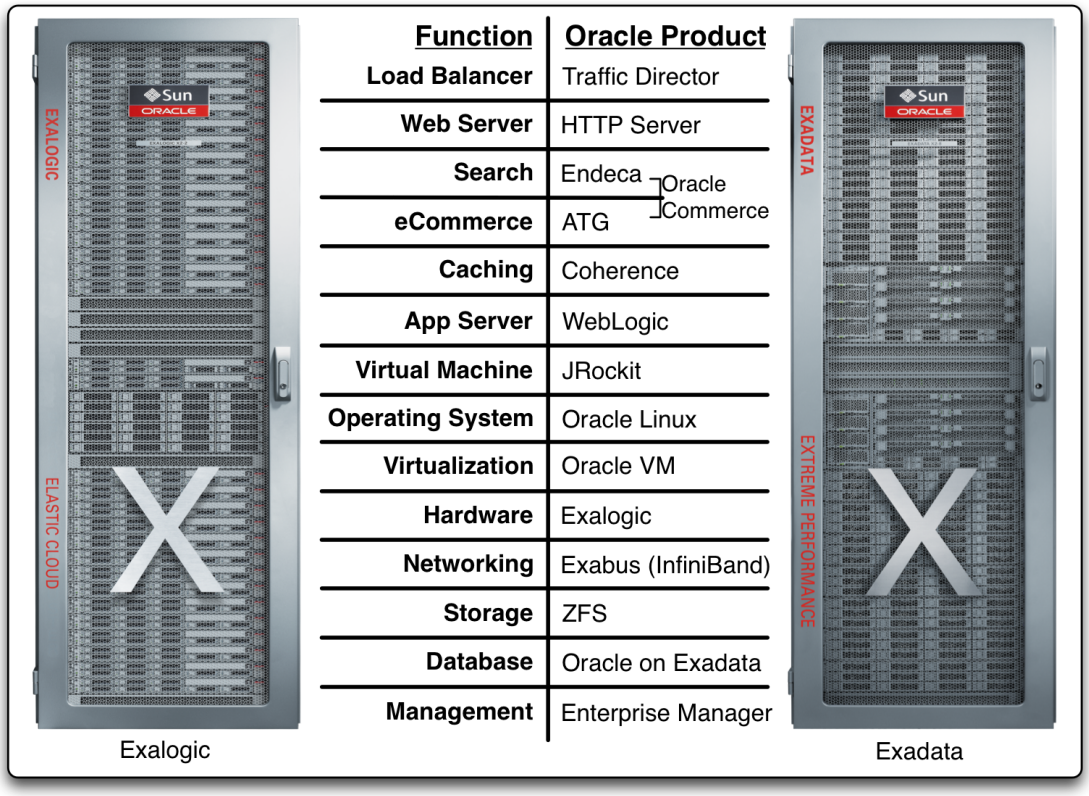


Figure 1

Oracle owns all products in the solution and has invested substantial resources to ensure that all of the solution components work exceptionally well together. While this white paper features Oracle's eCommerce solution, Oracle Commerce, as the application, most of the content is applicable to any

[1] 2011 Internet Retailer Top 500 Guide

application deployed to a Java EE-compliant container. Heterogeneous environments are a normal part of enterprise architecture.

The key to scalability is choosing a partner that provides the technology, services, and support required to implement eCommerce. Oracle is that partner and this white paper will describe Oracle's offerings and the architecture principles required to succeed.

Introduction

Over the past decade, eCommerce has evolved from a single application supporting a single channel to an ecosystem of related applications, middleware, and core technology supporting an increasingly converged set of all physical and virtual channels. For example, many trips to a physical store now begin with online research. For some Oracle customers, the web is the sole channel. Below is a depiction of all the touch points many organizations now have with their customers:

Figure 2

As a result, many purchases today, even those that take place in-store, are driven by the eCommerce channel. In fact, a 2011 study[2] showed that 48% of all retail sales are either online purchases or web-influenced purchases. Oracle technology can power the entire eCommerce ecosystem, from in-store point-of-sale systems, to warehouse and transportation management systems and beyond.

[2] http://www.marketingcharts.com/direct/web-influences-half-of-retail-sales-19730/

As a result of these additional touch points, the traffic these systems need to support has grown exponentially. For many, the cost of downtime leads to front-page newspaper articles, loss of revenue, and loss in shareholder equity. Therefore, it is imperative that scalability be a key design principle.

Oracle's eCommerce Platform Solution

Oracle offers a complete integrated suite of products for eCommerce, with the ability to leverage all of the pre-integrated solution or any combination of products. Specific products include:

Oracle Traffic Director is a robust and scalable software-based load balancer that is built into Oracle Exalogic, Oracle's hardware platform for Oracle applications. Oracle Traffic Director eliminates the need to use web servers for load balancing, which makes it easier to scale up or down based on real-time demand. Because it's hardware-accelerated and natively leverages the InfiniBand network found in Oracle Exalogic, the performance is unbeatable

Oracle HTTP Server is a web server based on the Apache HTTP server. It sits behind Oracle Traffic Director and is responsible for serving static content (e.g. images, JavaScript files, CSS files, etc) up to a Content Delivery Network (CDN)

Oracle Coherence is an in-memory distributed data grid solution, providing the ability to access terabytes of data within microseconds. Applications, including ATG, can store plain old Java objects (POJOs) and database objects as represented by various object relational mapping frameworks (ORMs). Storing objects in Oracle Coherence improves performance and scalability through eliminating load on the database and other systems

Oracle WebLogic provides a fast, reliable, secure, and powerful container for Java EE applications. It has tight integrations with Oracle JRockit, Oracle Database, and Oracle's engineered system for Oracle middleware and Applications, Oracle Exalogic

Oracle JRockit is the industry's fastest JVM, providing a stable runtime for Java-based applications, deterministic performance, and numerous tooling options

Oracle Exalogic is a rack-based system containing compute nodes (x86 servers), RAM, FlashFire SSDs, and ZFS storage, all pre-integrated to work well together and connected using Exabus (InfiniBand networking technology and related protocols). Each configuration of Exalogic contains the appropriate amount of RAM, SSD, and storage so that the system is "balanced" for optimal performance

Oracle Exadata is a rack-based system containing compute nodes (x86 servers), RAM, PCIe-attached Flash, and storage, all pre-integrated and optimized for the purpose of running Oracle Database exceptionally well. Numerous changes have been made to Oracle Database to better support the hardware, so that Exadata delivers performance and scalability that is unattainable through other means

Oracle Enterprise Manager is a single fully integrated apps-to-disk management and monitoring solution, for both Oracle and non-Oracle products. It offers complete lifecycle management for all Oracle products, while improving SLAs and deployment flexibility

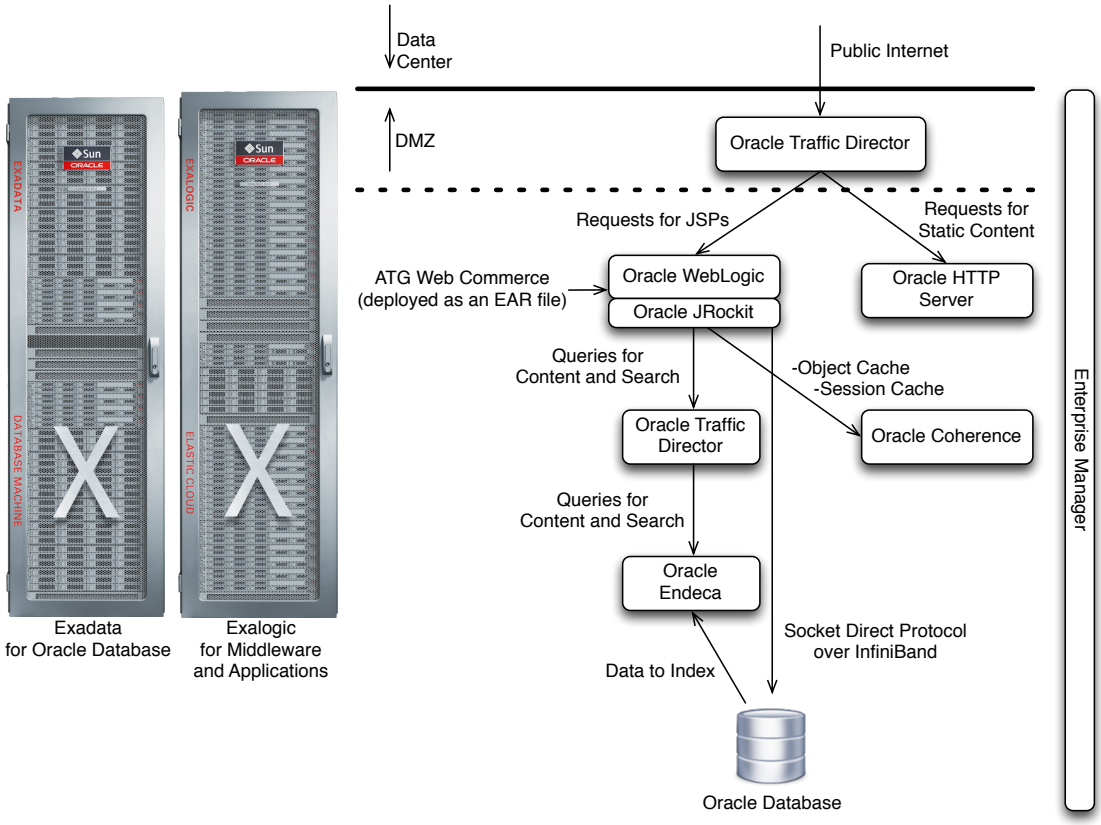
Oracle ATG Web Commerce, or simply ATG, is Oracle's eCommerce platform. It's a complete Java-based platform that allows you to deliver a personalized shopping experience across all customer touch points, including web, kiosk, call center, social media, and mobile

Oracle Endeca Guided Search complements ATG by providing search and guided navigation. Based on search queries, input from business users, advanced customer segmentation, and third party data (e.g. sales, social, analytics), Endeca provides search results and the ability to navigate through the results through faceted navigation

Oracle Endeca Experience Manager allows business users to dynamically assemble page layouts and page content for different customers, leveraging data from many different sources

Oracle Commerce is Oracle's solution for eCommerce, comprised of Oracle ATG Web Commerce + Oracle Endeca Guided Search + Oracle Endeca Experience Manager, all pre-integrated

The products come together to form the following architecture:



What Is Scalability?

Strictly speaking, scalability is the ability of a system to increase its throughput (for example, concurrent users supported) by adding resources (for example, hardware). The desired goal is to increase throughput in perfect linear proportion to the amount of resources added to a system. One unit of output should equal one unit of input. Two forms of scalability will be discussed: vertical and horizontal scalability.

Vertical Scalability

Vertical scalability is the ability to gain marginal throughput by increasing the physical resources of a server (like CPU or memory). An example of vertically scaling in the database tier would be purchasing a more powerful server (better CPU, more memory) to run a single Oracle RAC node. This is in contrast to horizontal scalability, where the throughput increase would be gained by adding an additional Oracle RAC node on a new physical server, or even by building a separate database.

If the marginal input (e.g. hardware) equals the marginal output (e.g. throughput in terms of page views/second/core), the resource is considered perfectly linearly scalable. A resource can have very low throughput but still be perfectly linearly scalable. A resource is no longer perfectly scalable when the marginal input no longer equals the marginal output. It is important to not confuse low throughput with poor scalability. The two are separate, but related.

Some resources must be vertically scaled. For example, legacy systems that Oracle Commerce integrates with sometimes must be vertically scaled. This is particularly true of some order management systems that cannot scale horizontally to cope with the increase in orders coming from a website during peak shopping periods like Cyber Monday.

While vertical scaling can increase overall throughput, limited redundancy can sometimes be a negative consequence. If a system relied solely on vertical scalability to deliver the required system throughput, the failure of the single node would jeopardize the stability and availability of the entire system.

Horizontal Scalability

Horizontal scalability is the ability to gain marginal throughput by adding more resources (usually physical servers), as opposed to increasing the capabilities of an existing resource (usually a physical server). An example of horizontally scaling a resource would be adding an additional Oracle RAC node to a database, as opposed to increasing the processing power of the hardware running an existing Oracle RAC node.

Oracle's platform for eCommerce is designed to be horizontally scaled. As the system needs to handle more sessions or more HTTP requests, hardware may be added until the system is capable of meeting the additional demands. Gaining additional overall throughput is generally performed by adding more instances of a given application, as opposed to increasing the throughput of an existing instance. Oracle's products have exhibited near-perfect horizontal linear scaling in lab tests and in live production environments.

While there are some special considerations for large-scale eCommerce, the solution scales linearly in part because it can be configured to not use any shared resources. Shared resources are single points of failure that can become bottlenecks. Shared resources are typically single operating system processes, like a web service hosted on a single JVM.

To achieve truly infinite horizontal scaling, a system can have no shared resources. This is known as a "shared nothing" architecture. An entire system is only as scalable as its least scalable resource. Work has to be done to remove potential bottlenecks, both by changing how the application interacts with shared resources as well as working to make those shared resources horizontally scalable.

Throughput

Throughput refers to the amount of work or capacity that a system or resource can perform or handle. Building a large-scale application ultimately requires a very high overall throughput – far more than can be delivered with a single server. As such, horizontal scalability is required to ensure enough capacity to serve the traffic of a large-scale application. It is important not to confuse throughput with overall system-wide scalability - the two are independent of each other. A JVM that can serve five pages per second may be more scalable than a JVM that can serve 50 pages per second. There is no causal relationship between low throughput and low scalability. If the marginal input (e.g. hardware) equals the marginal output (e.g. throughput), the resource as a whole is considered perfectly scalable.

While throughput is not directly related to scalability, scalable resources tend to have high throughput. High throughput also reduces the need to scale resources and generally leads to lower total cost of ownership. Going back to the previous example, a single instance that can serve 50 pages per second allows less physical hardware to be used than the lower throughput system. Reduced marginal throughput is usually the first sign of a scalability problem, so it's important to track it over time for each resource.

While throughput is not directly related to scalability, low throughput can indirectly lead to scaling problems. For example, a third party system may only be able to open 512 socket connections due to constraints in the operating system on which it runs. If each JVM connecting to this system consumes a socket, the maximum number of JVMs is 512. If throughput is half of what it should be, the problem will be encountered 2x faster than otherwise.

Building a large-scale e-commerce application requires a combination of the best throughput attainable with careful design to ensure individual resources are also highly scalable.

Scalability Best Practices

Overview

Scalability is a key consideration for any large-scale eCommerce platform. The following five rules will help to ensure the best and most scalable platform:

1. Decouple: Use shared resources as sparingly as possible
2. Cache Intelligently: Cache as much as possible as close to the end-user as possible

3. Leverage Converged Infrastructure: Oracle's engineered systems are perfect for eCommerce
4. Build in Redundancy: Confidently handle failures and bursts of traffic
5. Plan and Collaborate: Build your platform to meet stakeholders' goals

Each of these will be discussed in greater detail.

Rule #1: Decouple

Shared resources are single points of failure that can become bottlenecks. Shared resources are typically single operating system processes, like a web service hosted on a single JVM. If a remote call is made synchronously, the system being called into is considered a shared resource because its failure would jeopardize the availability of an entire system.

Traditionally, all processing in an e-commerce application was performed synchronously. Examples of synchronous processing include submitting an order to an order management system, retrieving inventory from a third party system, or sending a customer a marketing email. With some good design and a little bit of work, all these activities and others could be deferred until a later time, thus minimizing the use of shared resources. Synchronous processing is fine for smaller websites, but it leads to three major problems in larger websites.

The first problem is that the system must scale to meet the demands of peak customer traffic that must be handled plus all of the non-essential processing that could be deferred. During peak loads, the system should only be doing essential (revenue-generating) processing. In other words, the system should only be servicing customer requests and taking orders. The non-essential processing should be dropped in a queue and performed when the system is more lightly loaded, like during the middle of the night.

Remember that some shared resources (especially legacy systems) are not at all scalable beyond a certain point or are prohibitively expensive and/or time-consuming to scale. Scaling for only essential processing helps prevent the need to scale those systems.

The second problem with synchronous processing is that you don't have the flexibility of designating when, where, or how the work is to be performed. If non-essential processing is dumped into a persistent queue, it can be executed anywhere at any time, with different priorities given to different types of jobs. For example, nobody would argue that archiving older orders should take priority over accepting new ones. The processing that is more important and time-sensitive can be scheduled to run ahead of the other less important processing.

With the work in a persistent queue, it can be performed anywhere – on one instance dedicated to a specific function, to a cluster of instances dedicated to a specific function, or on each instance in a production environment. For example, orders may be submitted to an order management system on each instance in a production environment whereas you may have a dedicated cluster of instances for purging orders.

The third and biggest problem with synchronous processing is that the entire system's scalability is bounded by the limits of the least-scalable resource. For example, an SMTP server with limited

scalability may be used to send out all emails. If order update emails are sent synchronously as the order management system updates orders, the SMTP server could become overloaded. But if the emails were dropped in a queue and processed at a rate the SMTP server can manage, the SMTP server wouldn't be forced to scale beyond what is absolutely required. The work would be evenly distributed over time.

Oracle Commerce ships with a robust set of intra-JVM queuing tools. These tools allow synchronous processing to be made asynchronous, but only within one JVM. Functionality to persist work to a database and process it on another instance is requires custom code that is easy to implement. WebLogic also ships with incredibly robust JMS functionality that makes it simple to defer processing by leveraging queues.

Rule #2: Cache Intelligently

It's important to cache as much as possible as close to the end-user as possible in order to improve server-side response times, increase the vertical scalability of each JVM, and reduce load on resources like databases.

eCommerce applications are characterized by extremely dynamic, personalized pages, which can make caching more difficult. Fortunately, Oracle provides numerous caching mechanisms to intelligently cache data in the application-tier.

Content Delivery Networks (CDNs) prevent most HTTP requests from ever reaching your infrastructure. Instead, requests for static media (e.g. image files, Java Script files, etc), and increasingly, entire pages, are served directly from "edge" servers that are geographically close to the end-customer. The number of HTTP requests serviced by a CDN can greatly outnumber the HTTP requests serviced by your application, so the impact of using a CDN can be significant. Requests that the CDN services are requests that your system doesn't have to scale for.

Rule #3: Leverage Converged Infrastructure

The importance of physical infrastructure cannot be under-stated in a large-scale eCommerce environment. Recognizing the importance of infrastructure, Oracle has made substantial investments in "engineered systems," which are fully integrated stacks of hardware and software that are optimized to work well together. The vertically-integrated nature of Oracle's engineered systems brings unmatched scalability, manageability, and time-to-market. Oracle Exalogic Elastic Cloud (or simply Exalogic) is Oracle's engineered system for the middle and application tier.

Exalogic is a rack-based system containing compute nodes (x86 servers), RAM, FlashFire SSD, and ZFS storage, all pre-integrated to work well together and connected using Exabus (InfiniBand networking technology and related protocols). Exalogic is available in configurations containing 64 CPU cores (⅛th rack), 128 CPU cores (¼th rack), 256 CPU cores (½ rack), and 480 CPU cores (full rack), with the ability to seamlessly link up to eight racks of Exalogic together using Exabus.

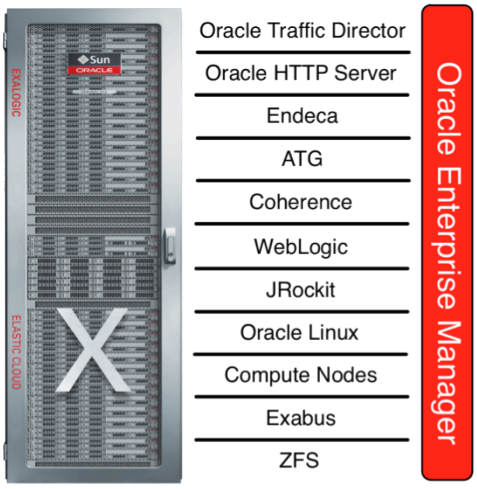


Figure 3

Each configuration of Exalogic contains the appropriate amount of RAM, SSD, and storage so that the system is "balanced" for optimal performance. Configurations below a full rack (⅛th, ¼th, ½) may be upgraded (e.g. ¼th to ½) without any downtime. Each configuration leverages the same physical rack.

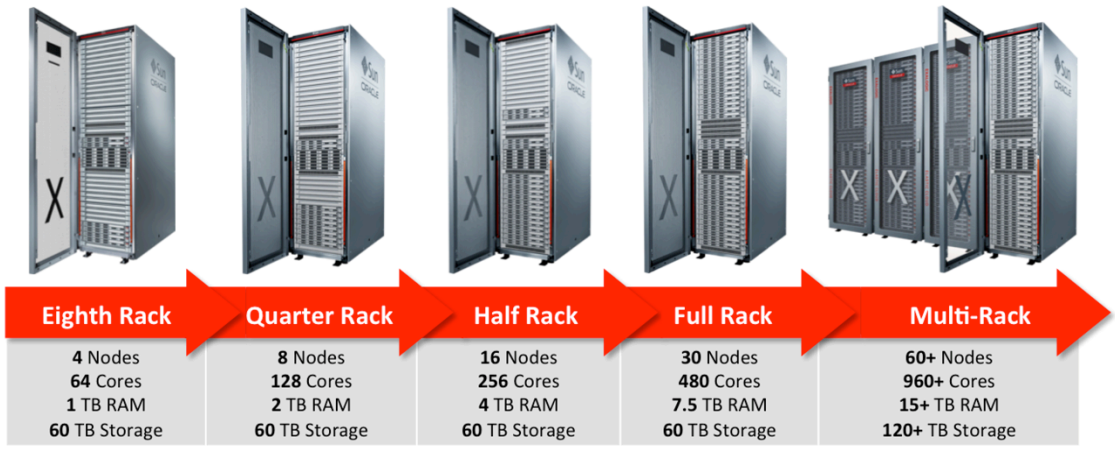


Figure 4

Oracle Linux or Solaris 11 for x86 may be selected for the operating system, with each having been extensively tuned for the underlying hardware. Oracle Linux, like Red Hat Linux, is based on the Fedora Core codebase, so custom non-Java applications are likely to be highly compatible. Oracle's JVMs (HotSpot and JRockit), Oracle WebLogic, Oracle Coherence, and many Oracle applications have been enhanced and tuned to transparently take advantage of the hardware and software below it in the stack. The additional changes and tuning are fully transparent, so no special knowledge or hotfixes are required. This vertical integration between software and hardware is part of what enables Exalogic to provide such exceptional performance for eCommerce websites. Similar gains can be realized in the database tier by using Exadata Database Machine, an engineered system optimized for OLTP and OLAP workloads.

A defining feature of Exalogic is the elimination of I/O bottlenecks through an I/O subsystem called Exabus. This subsystem is a collection of technology including InfiniBand switches, gateways, host channel adapters, firmware, device drivers, operating system extensions and software libraries. Together, this technology allows the kernel and operating system's TCP/IP stack to be bypassed (also known as Remote Direct Memory Access, or RDMA) for most inter-process communication. Within the same Java process, I/O bottlenecks are eliminated through extensive tuning at all layers.

Exalogic is engineered to be managed and monitored as one single stack. Oracle Enterprise Manager (for software) and Oracle OpsCenter (for hardware) allow comprehensive system-wide management because they were modified and configured for Exalogic. While Enterprise Manager and OpsCenter work well in a heterogeneous environment with non-Oracle products, they work especially well with Oracle products including Exalogic. Patching and other maintenance becomes a lot easier because Oracle can provide single file patches (from storage to operating system) due to its knowledge of each system's configuration. With a finite and well-known number of system configurations, it becomes easy for Oracle to release consolidated patches. Finally, embedded hardware diagnostic capabilities allow for Exalogic to "phone home" to create Oracle Service Requests with Oracle Support in the case of hardware failures. The integrated nature of Exalogic, the quality of the products on their own, and the value of the integrations between these best-of-breed products inside of Exalogic allows for unparalleled management, monitoring, performance and ease of maintenance.

Internally, Oracle product engineering (including ATG) uses Exalogic for performance testing, QA testing, and other times when hardware is required. Oracle also uses Exalogic as the foundation for its Cloud. Exalogic is easy to set up and performs exceptionally well, which makes it optimal for an environment such as Oracle's development organization.

Exalogic and Exadata are conceptually similar, with both sharing the same InfiniBand networking stack and other core innovations. The major difference being that Exadata is built for running the Oracle Database whereas Exalogic is built for running Oracle middleware and applications. Exadata is the preferred platform for the database(s) behind eCommerce apps because of its performance, reliability, advanced connection capabilities to Exalogic, and overall lower total cost of ownership. Any application certified on WebLogic is automatically is fully certified to run with Exadata and requires no further optimizations to take advantage of Exadata.

Exadata comes in quarter, half, and full rack configurations. A full rack of the X3-2 model contains 256 CPU cores, 1 TB of RAM, 14 storage cells (up to 224 TB total usable storage), and 22.4 TB of PCI-based flash. A full rack of Exadata is far more than sufficient for most eCommerce websites. Within one rack of Exadata, there may exist multiple database instances, each of which may or may not be in a clustered RAC configuration. A single database may be split across two or more racks of Exadata in a RAC configuration, for high availability. A database on Exadata may be replicated to another database for disaster recovery using Oracle Data Guard. Exadata runs Oracle Database 11g Release 2, but with transparent optimizations for the Exadata hardware and the Oracle Database software. Any feature available in Oracle Database 11g Release 2 is available on Exadata.

Exadata can run the Oracle Database exceptionally well for the same reasons that Exalogic runs Oracle middleware exceptionally well – extensive tuning, engineering the hardware and software to work

together, and innovations that are only available with the stack. Exadata contains a number of innovations, including:

- **Hybrid Columnar Compression:** Traditionally, all columns for a particular row are stored sequentially within a single database block. This allows for fast record-oriented read access but allows for only minimal compression. The alternative is to store columns of data together, which allows for high compression but can create excessive I/O for multi-column access. Hybrid Columnar Compression permits data to be stored in a hybrid of both, which achieves the benefits of columnar storage compression and the performance of sequential row storage. Average storage savings are 10x-15x with some customers seeing significantly higher compression rates, depending on the data structure. So the 224 TB of usable storage in a full rack is effectively 2.24 PB (assuming 10x compression)
- **Smart Scans:** In a non-Exadata configuration, an Oracle database typically uses a SAN for storage. When a query is executed, all relevant rows and columns are returned to the database, with the database CPU performing the filtering and returning the resultset to the application. With Exadata, each storage cell contains 16 CPU cores and portions of the Oracle Database software, with the ability to perform data filtering at each storage cell. As a result, the database receives only the data it actually needs and applicable queries will be executed in parallel across each of the 14 storage cells. This results in less data being sent to the database and less processing needed in the compute nodes
- **Smart Flash Cache:** A full rack of Exadata contains 22.4 TB of flash (PCIe cards, not flash disks), which functions as an intelligent cache to offload physical I/O from the disks. This is a particularly beneficial feature for eCommerce applications, which often perform the same queries over and over again. Smart Flash Cache allows up to 1.5 million random I/O operations per second (IOPS) and can scan data at up to 75 GB/sec. This feature allows for at least 10x better performance (roughly ½ a millisecond per single block read) when compared to a traditional disk
- **Smart Flash Logging:** The redo logs that would traditionally be written solely to disk may be optionally committed to flash first, with eventual copying to disk happening asynchronously
- **Full Database Encryption:** Hardware-based encryption may be used to encrypt a database running on Exadata. This is particularly important for eCommerce websites, where a database may store credit cards and other Personally Identifiable Information (PII). Moving encryption and decryption from software to hardware results in a 5x performance improvement. Data can be decrypted at a rate of hundreds of gigabytes per second with virtually no overhead

While Exadata and Exalogic are outstanding on their own, they are even more powerful when linked together. Using the Active GridLink for Oracle RAC feature found in Oracle WebLogic, an application running on Exalogic can communicate with an Oracle database running on Exadata using Exabus, at a rate of 960 gigabits/sec with latency of 1.2 microseconds. To put that into perspective, 25.5 full-length uncompressed DVDs could be transferred between Exalogic and Exadata in a single second. Traditionally, application servers are connected to databases over gigabit Ethernet with milliseconds of latency. In addition to raw throughput and low latency, Exabus allows the TCP/IP stacks (and thus

kernels) to be bypassed in both Exadata and Exalogic. Together, these optimizations allow for 3x better OLTP performance.

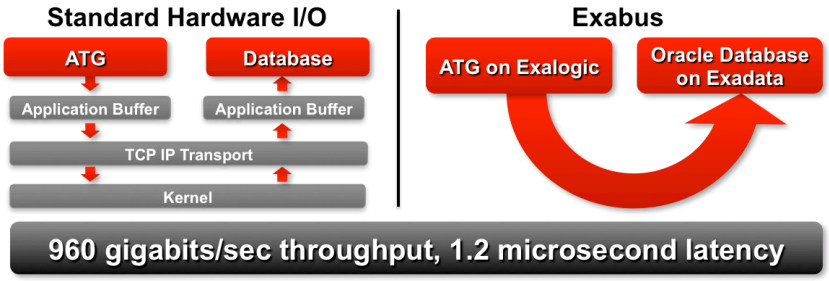


Figure 5

In addition to latency and throughput improvements, Active GridLink for Oracle RAC provides other functionality such as fast connection failover, runtime connection load balancing (balance queries across RAC nodes based on real-time load), and XA affinity (bind certain queries back to the same RAC node).

Rule #4: Build in Redundancy

Increasingly, eCommerce websites are being served from two or more data centers to provide maximum availability. These data centers range from a few miles apart to a few thousand miles apart. The second (or n^{th}) data center may serve as a virtual safe deposit box for real-time database backups, or on the opposite side of the spectrum, a fully-functional data center, capable of handling all of production traffic on its own. The level to which the second (or n^{th}) data center is used depends on the amount of acceptable downtime (Recovery Time Objective or RTO), the amount of acceptable data loss if a failure occurs (Recovery Point Objective or RPO), and finally the ability of an organization to execute (financial resources, competency, time, etc).

For many organizations, the cost of even a brief unexpected outage or planned downtime far outweighs the cost of setting up one or more additional data centers. For example, average annual online revenue for the top 100 eCommerce websites in the U.S. was $1.58 billion[3]. At $1.58 billion, an hour of planned or unplanned downtime costs an average of $366,000. Since unplanned downtime is more common during traffic peaks, the real cost of downtime is often substantially higher. At 99.5% uptime (allowing for 50.4 minutes per week of downtime for builds and unplanned outages), the lost revenue comes out to $7.8 million over the course of a year in just planned downtime.

In addition to ensuring maximum availability, other reasons for operating from multiple data centers include:

[3] 2012 Internet Retailer Top 500 Guide

- **Better Fault Tolerance:** The impact of natural disasters (e.g. fires, tornados, earthquakes), human error (e.g. cable cuts, misconfiguration), and equipment failure (e.g. networking, database, storage) and other causes of unexpected down-time are likely to be confined to one data center. Maintaining adequate physical distance between the data centers and adhering to the principles outlined in this document all help to ensure that faults will be isolated and that business continuity will be ensured.
- **Better Performance:** Having data centers close to customers ensures the best possible performance by removing latency. This is the concept behind a CDN. For example, round-trip latency between London and Tokyo over the public internet is 440 milliseconds. With a data center in Tokyo, the latency for customers in Asia would be a few dozen milliseconds at most. The performance improvement would be noticeable, especially when Ajax-heavy pages are used. Good design principles, careful coding and full use of a CDN or appliance-based application accelerators minimizes the impact of latency but does not eliminate it.
- **Easier Releases/Maintenance:** With two or more autonomous data centers, routine code releases and ongoing maintenance can become a lot easier and can result in less planned downtime. While it is possible to avoid down-time if one data center is used, having two or more data centers allows an entire data center to be taken offline for the duration of the code release or maintenance, and required testing. Having a replica of the production database offers many opportunities to reduce or eliminate planned downtime. Also, sessions can be failed over to another data center in near real-time using Oracle's Coherence*Web feature or sessions can be bled (also known as "drained") using a CDN's capabilities.
- **Extra Capacity:** In periods of exceptionally high demand, such as Cyber Monday, it would be beneficial to have the capacity of two or more data centers should the capacity of one data center not be enough. Some data centers have limited floor space or the cost per square foot is excessive

Operating from more than one data center is not without its challenges. Due to the unique deployment architectures, requirements, legacy systems, and resource constraints of each organization, there is not a single prescribed methodology. There are however numerous design patterns, tools, and products that can help. The discussions in this area tend to be around "how," not "if." Often, elements of these various approaches can be combined to create entirely new approaches.

The major issue to overcome in any distributed environment is updates being made to the same record at the same time from two or more distant databases. Due to latency, data synchronization between the databases must often be asynchronous. With asynchronous data synchronization and no changes to configuration, the problem of data being overwritten or corrupted will always exist.

With most applications, this problem manifests itself when two or more customers log in using the same account from two separate data centers. Customers frequently share their credentials with family members or others or sign in from multiple devices (e.g. laptop and smart phone). When this happens, two or more customers share the same profile and orders in the database. The following diagram depicts what happens if two customers attempt to update the same profile at the same time from distant databases:

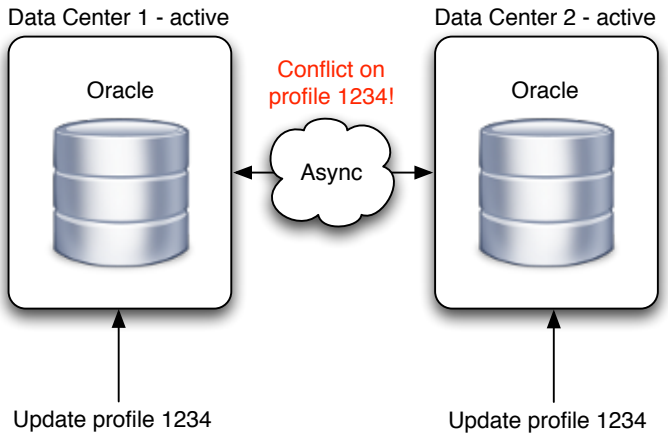


Figure 6

In an active/passive configuration where customers can only write to one database, this is not an issue. The following diagram depicts what happens if two customers attempt to update the same profile at the same time from one data center:

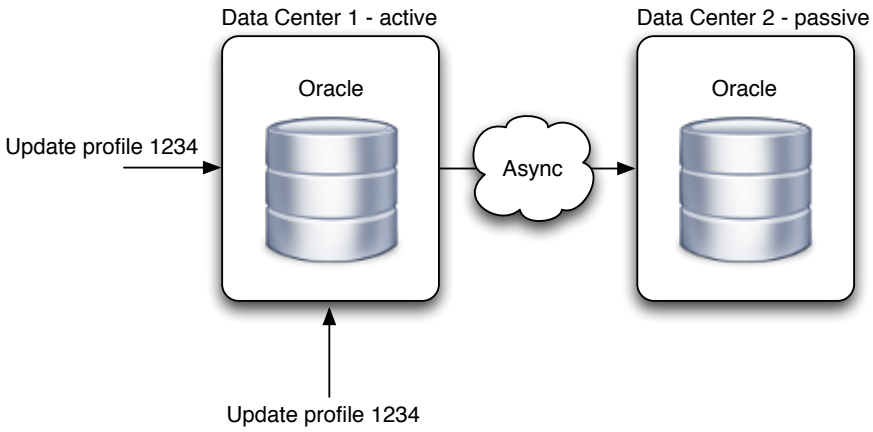


Figure 7

Accurate proximity-based load balancing (e.g. load balancing by continents, countries or regions) can reduce the frequency of occurrences. Most of the time, customers share credentials with people they know and are close to – spouses, family, friends, etc. These individuals tend to be relatively close together. With proximity-based load balancing, though it can happen, it is rare that two customers using the same credentials would access different data centers.

While the aforementioned scenario is rare, it is a problem that most organizations find unacceptable. For each of the approaches presented in this document, there are a series of changes that can be implemented to guarantee that this never happens.

The solution to this problem is to force all customers sharing the same credentials to write to the same database (though not necessarily same data center). With all updates occurring to the same database, it is impossible to have conflicts due to asynchronous data replication. Changes can be made to elements

of the stack to make this happen. All solutions must be balanced against the desired RTO, RPO and the ability of an organization to execute.

The solutions presented include:

- **Using a standby data center and standby database:** With all updates occurring to the same database, conflicts are not possible.

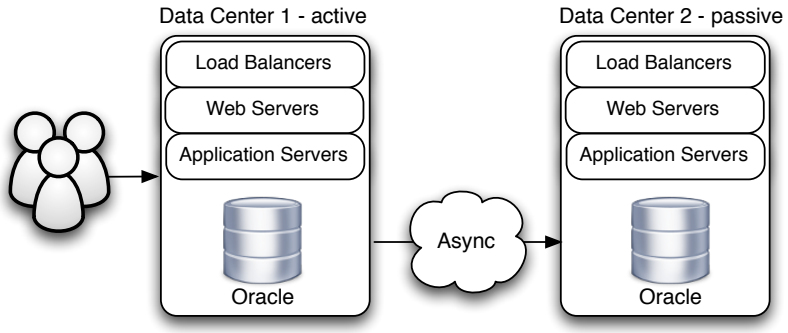


Figure 8

- **Writing to databases in different data centers:** If latency between the databases is not too great (roughly 500 miles separating the data centers), a database in an alternate data center can be used if your eCommerce application determines that the user is already logged into a different data center. Even if customers sharing the same account credentials are in different data centers, their database operations will all be confined to one database. Only customers that are deemed to be in the "wrong" data center will use a database in a different data center. This is the basis for the "Active/Active (fewer than 500 miles separating the data centers)" approach below.

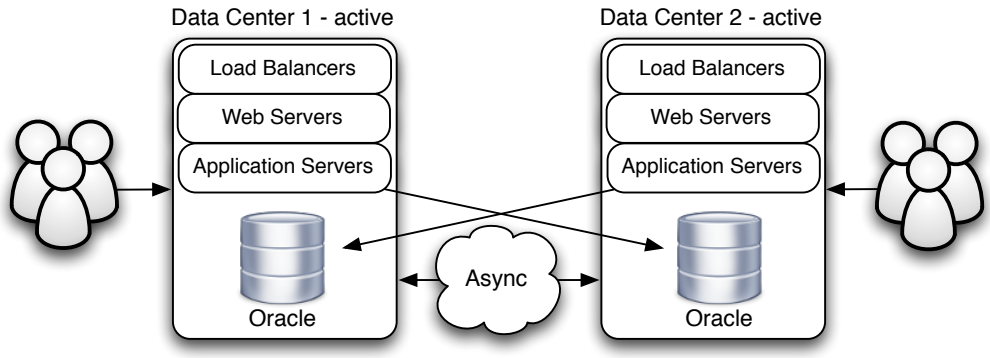


Figure 9

- **Redirecting customers to a different data center on sign-in:** When the customer signs in, your application should be configured to recognize if the customer already has an active session in a different data center. If so the customer will be permanently redirected to the data center containing active sessions for a given account.

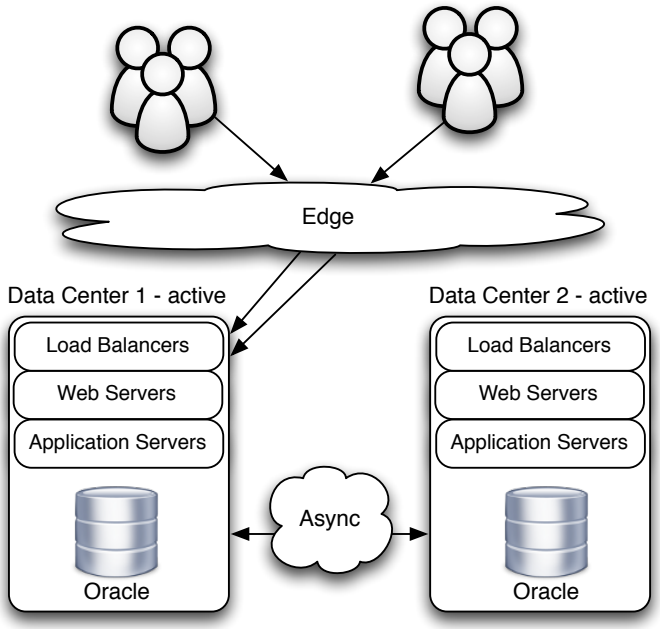


Figure 10

More information about these approaches can be found at http://www.oracle.com/us/products/applications/atg/architecting-oracle-atg-518346.pdf

Rule #5: Plan and Collaborate

Involvement from business stakeholders should go beyond requirements gathering and user acceptance testing. Getting involvement from business stakeholders throughout design, implementation and continuing through the maintenance phases is important for any large-scale eCommerce deployment. Collaboration is important for any deployment, but larger deployments require even more collaboration due to the nature of the decisions that must be made by both sides.

Business users may not fully understand the technical implications of decisions that they make. For example, business users may set a very demanding internal service level agreement (SLA) but not understand the technical and financial implications of that decision. There are also decisions that technical administrators must make that impact business users. For example, technical administrators may disable anonymous order persistence to save database space and unintentionally limit the ability to obtain revenue that might be realized by persisting orders. Decisions by both parties must be made collaboratively to attain the best results.

Below are a few examples of decisions that need to be made collaboratively:

- **Internal SLAs:** It is common for business users to set demanding SLAs without providing the funding required to meet the SLA. The way to meet demanding SLAs is by leveraging Oracle's engineered systems and by building redundancy into your system, from disk drives all the way through to operating multiple geographically-disperse data centers. Fully redundant

environments are expensive and business users need to be aware of the consequences of the SLAs that they want. The decision should be made jointly to find the right balance between higher service levels and costs

- **Project Budget Requirements:** Without thoroughly defining and properly sizing all of the environments at the start of an implementation, business users will become frustrated by never-ending hardware funding requests. All of the environments and all of the hardware that needs to be purchased should be worked out at the start of an implementation. Deciding what hardware to purchase is a collaborative effort
- **User limits:** Users, both human and search engine bots, may need to be blacklisted from a system if they abuse it. For example, 20 login attempts in one second from the same IP address is probably a script trying to hack into someone's account. For the stability of the system and the protection of customers, abusive users must be blocked. Business users need to work with technical staff to define criteria for being blacklisted
- **Throttling:** Throttling, or the act of preventing new sessions from being created, may need to be done in emergency situations to keep a system up. If your application is memory-bound, and has been shown to only handle X number of sessions during load testing before failing, basic throttling should be enabled at 95% design capacity or some other pre-defined point. Business users need to help define that point and provide guidelines that may be needed to prioritize which users are allowed to reach the site. Prioritized customer requests will be given priority over all other requests during a throttling situation, so it's important to spend time defining exactly what a prioritized customer is. Oracle Traffic Director provides these throttling capabilities

These are just a few examples of decisions that need to be made collaboratively. Everything is a tradeoff.

An area that the two sides must be in complete sync on is the management of traffic. Business users control all of the levers that drive traffic – campaigns, promotions, flash sales, etc. With advanced warning, system administrators can ensure all required capacity is available and ready for the surge in traffic. They can shift schedule maintenance to ensure it does not interfere with the business plans and may also want to supplement normal operations when appropriate. For example, they might ensure additional monitoring is put into place and that key personnel are available and on call to ensure that everything works well during the heavy load.

Not all traffic is anticipated. Some is unanticipated. For example, a mis-placed decimal point in a marketing email can make a 10% off coupon 100% off, which may result in a tremendous and unanticipated surge in traffic. Unless there's thorough quality control, there will be problems like this.

In summary, both business stakeholders and technical staff need to work closely together for the duration of the software development life cycle.

Sizing

Properly sizing an environment is something that must be given extra attention for a large production deployment. This includes selecting the size of Exalogic and Exadata units to purchase or in the case of a commodity-based system, figuring out which servers, processors, storage, memory, etc to purchase. A smaller deployment may have 50% extra capacity with very minimal cost because hardware is typically acquired in larger size units than is actually required. Many smaller websites actually do this because it's relatively inexpensive and it reduces risk of performance issues during unanticipated traffic spikes. With a large deployment, the sizing numbers must be much more accurate. An environment consisting of hundreds of CPU cores that is improperly sized by as much as 20% could lead to a lot of unnecessary cost.

To properly size a new deployment, it's important that realistic projections be obtained from business stakeholders. The entire system must be able to handle peak traffic, even if that only occurs a couple of days throughout the course of a year. Projections should include:

- Average & peak concurrent HTTP sessions
- Average & peak page views per second
- Average & peak orders per second
- Average & peak searches per second
- Average number of line items per order

Averages are important to collect, as you may elect to run your site with a reduced number of instances during average traffic and then add more instances to meet demand during the holidays or other periods of high traffic.

When sizing an environment, it's important to know what your "safety factor" is. Your safety factor is the percentage of your system's capacity that is not utilized at your projected peak as a percentage of the projected peak. For example, if the number of sessions you want to be able to concurrently service is 1 million and you have a safety factor of 25%, your system should be sized to be able to handle 1.25 million sessions. The safety factor is a fluid number that fluctuates with seasonal variations in traffic. The safety factor should take into consideration any promotional or unforeseen peaks of traffic. For example, a retail chain may send an email to all of their customers with a promotion, such as 30% off all orders + free shipping. That promotion could drive a lot of traffic to their site very quickly.

The safety factor decision should be made jointly between business and technology. Business owners need to understand the implications of the decision. Historically, Oracle's customers have used safety factors of at least 50%, with larger safety factors being mostly for smaller deployments.

Human Resources

Successful development, deployment and ongoing maintenance requires the right technical resources, regardless of the size of the website. This is even more true for larger websites, as specialized knowledge is required to deal with that scale and there is less margin for error. It's important to have a core of knowledgeable and well-rounded resources. Ideally, all technical resources should have experience with designing, developing, and deploying large-scale websites.

In addition to high quality application/deployment architects, expert DBAs are absolutely critical to a successful large-scale deployment. Since databases can quickly become a bottleneck, it's crucial that there are knowledgeable DBAs available. It's also important to find DBAs with experience in horizontally scaling databases and related technologies like partitioning and clustering. Smaller websites often use one database on one or two physical servers, but this is not sufficient for a very large site. DBAs should be supported by experienced network and infrastructure personnel, as large database clusters impose unique challenges to the network and storage infrastructure.

Supporting Architecture

Now that theory and rules have been discussed, let's review how Oracle's products can help out at each tier.

Web Tier

Over the past decade, the web tier has substantially changed in ways that improve scalability, the ability to provision, and security. The web tier includes the following components:

- Content Delivery Networks (CDNs)
- Load Balancers
- Web Servers

Together, these components must provide the following functionality:

- Load balancing
- Layer 7 security
- SSL termination
- Static content serving
- Throttling
- Reverse proxying

It's difficult to propose a prescriptive web tier architecture because CDNs, web servers and load balancers are each capable of providing the above functionality independently. Often functionality is duplicated across the tiers and the lines of delineation are blurred. Based on Oracle's experience in this space, the following architecture is recommended:

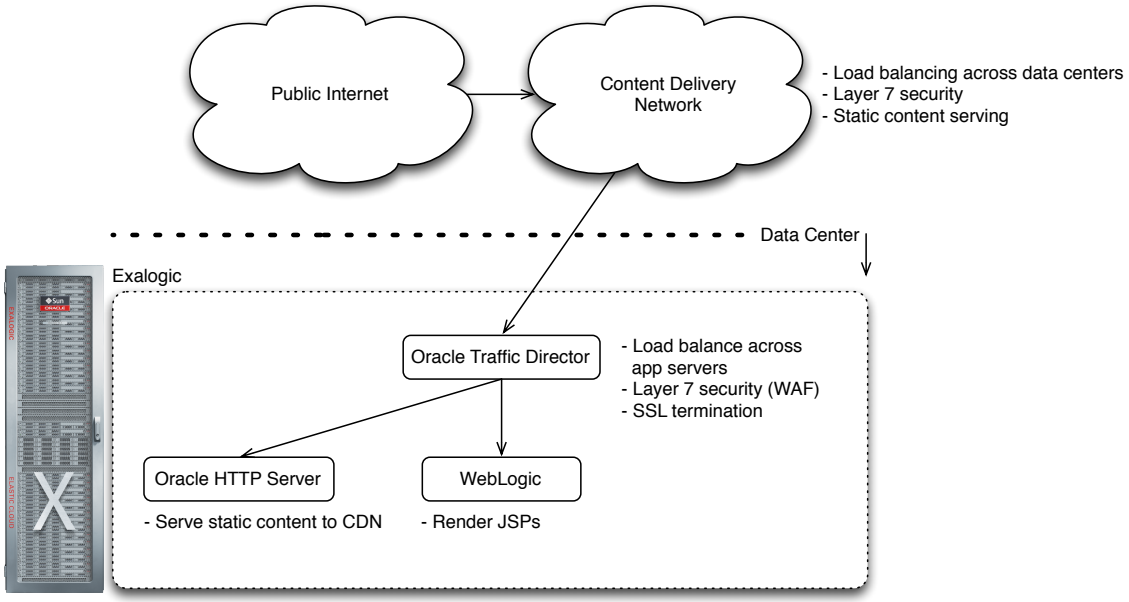


Figure 11

In this architecture, the CDN is acting as a giant distributed reverse proxy so that all HTTP requests must pass through it before going back to the origin data center. CDNs no longer just serve static content. Because CDNs are acting as a reverse proxy, they are able to take a more active role in load balancing, providing security (including protection against layer 7 attacks) and pre-fetching static content. CDNs are required today for large-scale eCommerce websites because of the weights of the pages being served and the number of HTTP requests generated for each page view. Single pages are typically over 1 megabyte and require over 100 HTTP requests. Serving websites to an audience on the other side of a continent or to other continents requires either local data centers or the use of a CDN acting in the capacity of a reverse proxy. Latency can quickly reduce page load times and CDNs are the primary means by which latency is reduced.

On-premise load balancers have traditionally been used to load balance across a farm of web servers, with the HTTP requests passing through the web servers prior to arriving at the application servers.

Load Balancing

Load balancing plays a crucial rule in all layers of any large-scale eCommerce environment. Load balancing is typically required to select:

- A data center which will service the end-customer's session (known as Global Site Load Balancing, or GSLB)
- A web server which will serve the static content
- An application server which will render JSPs and execute business logic
- A search engine which will respond to search queries, faceted navigation queries, and possibly more

A good load balancer is always available, scalable, fast, and secure. Oracle's primary offering is Oracle Traffic Director. Oracle Traffic Director, or simply OTD, is a robust and scalable software-based load balancer that is baked into Oracle Exalogic. Oracle Traffic Director eliminates the need to use web servers for load balancing, which makes it easier to scale up or down based on real-time demand. Because it's hardware-accelerated and natively leverages the InfiniBand network found in Oracle Exalogic, the performance is unbeatable.

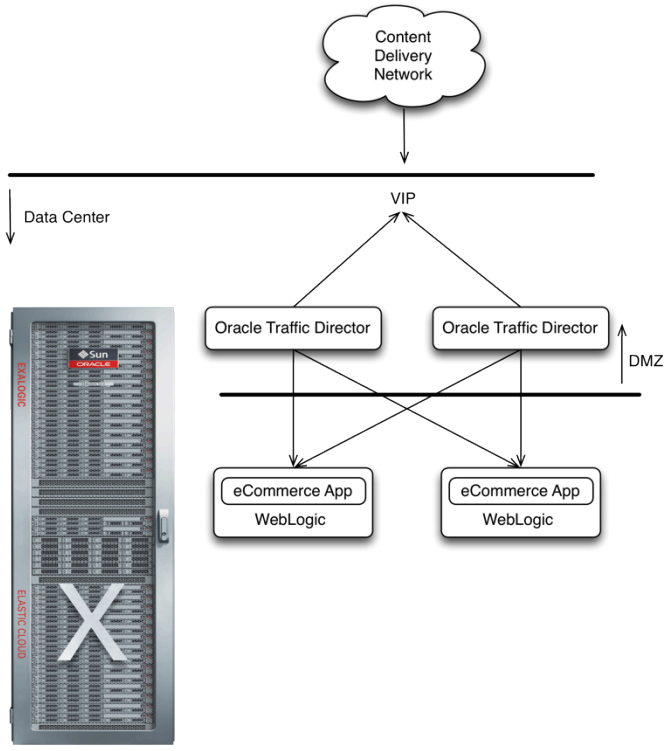


Figure 12

Oracle Traffic Director is only available on Exalogic and offers the following features:

- Optional use of InfiniBand over Socket Direct Protocol between Oracle Traffic Director and WebLogic
- Full built-in Web Application Firewall (WAF)
- Ability to scale out by allocating more hardware to Oracle Traffic Director
- Hardware-accelerated SSL encryption and decryption
- An easy-to-use management interface
- Comparable features to traditional hardware load balancers

Many customers leverage a traditional layer 7 hardware load balancer (e.g. Netscaler, F5, etc) in front of Oracle Traffic Director as part of a defense-in-depth security strategy. The hardware load balancer simply passes HTTP requests back to the single VIP that Oracle Traffic Director exposes but it's one

more layer that must be penetrated for an attack to succeed. It is recommended to use Oracle Traffic Director in place of traditional web server-based load balancing (e.g. mod_wl) in an Exalogic-based environment.

Web Server

Oracle's web server is called Oracle HTTP Server, which is based on Apache 2.2. It's essentially a wrapper around Apache but with some added functionality and full product support.

Traditionally web servers were in the DMZ and were used to:

- Expose multiple VIPs to the public internet
- Protect the application servers from attacks
- Load balance to application servers
- Serve static content

In the early days of the internet, web servers played a key role because CDNs were in their infancy and because the web was mostly static. Web servers were and still are very good at serving static pages. Because of their prevalence, they were given a central role as the web evolved and static websites became dynamic, with application servers and relational databases powering them. A typical architecture today is still as follows:

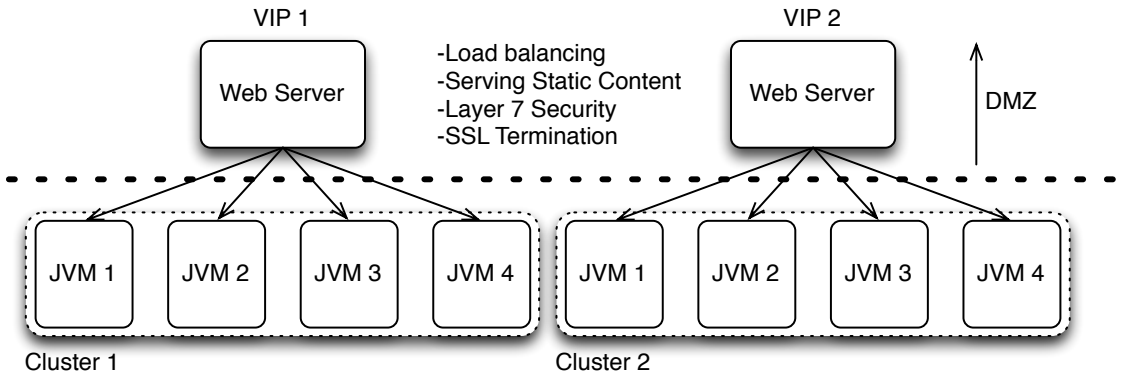


Figure 13

In this model, web servers are responsible for basically everything a load balancer is supposed to do, including SSL encryption/decryption, security, and load balancing. Each web server is "master" of a small cluster of instances. Web servers were never designed to be in this central role.

Over the past few years, the role of web servers has been greatly diminished because of the rise of CDNs and load balancers. For very good reasons, CDNs and load balancers often split up these tasks, which relegates web servers to serving up static content to a CDN.

The advantage of this model is primarily that each product is doing exactly what it's designed to do. The CDN is load balancing across data centers, providing a layer of security, and serving static content to end-users, all from data centers that are geographically near to end-users. Oracle Traffic Director is

directing requests for static content (content that the CDN has yet to cache) to a web server, directing requests for JSPs to WebLogic, terminating SSL, and providing a comprehensive layer of security.

Oracle Traffic Director is able to scale horizontally, simply by allocating more hardware to it. The same VIP is exposed to the CDN and the new instances see all of the other application server instances. It's a seamless way to dynamically scale up or down an environment. Scaling up an environment with just a web server would involve re-configuring DNS, adding a new web server, adding new application server instances under that new web server, and then testing it all to ensure it works properly. More hardware cannot simply be allocated for it to scale.

Application Server

Oracle WebLogic is the world's #1 highest revenue[4], fastest[5], and highest ranked[6] application server on the market today. It is because of WebLogic's technical and market strength that it serves as a core of Oracle's middleware suite, as well as the core of Oracle's platform for eCommerce. WebLogic is fully Java EE-compliant, leverages a modern modular architecture, is easy to manage, and is enterprise-ready so that it can support even the most demanding of eCommerce deployments.

Application servers play a key role in large-scale eCommerce deployments because of their role as the platform on which applications are deployed. As previously discussed, scalability is more than just whether a resource's output (e.g. HTTP requests/second) is equal to its input (e.g. CPU cores, RAM, WebLogic Managed Servers, etc). While WebLogic excels in that area, an application server's real value is its ability to be easily managed. A new application can be deployed to hundreds or even thousands of managed servers in a rolling manner with the click of a button. Configuration can be easily managed, audited, and diffed. New environments can be built from scratch in minutes. It's the management aspect of scalability where WebLogic truly excels.

---

[4] Source: Gartner, Market Share: All Software Markets, Worldwide – March 2012 – Based on total software revenue

[5] http://www.spec.org/jAppServer2010/results/

[6] http://www.gartner.com/technology/reprints.do?id=1-17GUO5Z&ct=110928&st=sb

Primarily because of how easy it is to manage, Gartner ranked[7] WebLogic at the top of its application server magic quadrant:

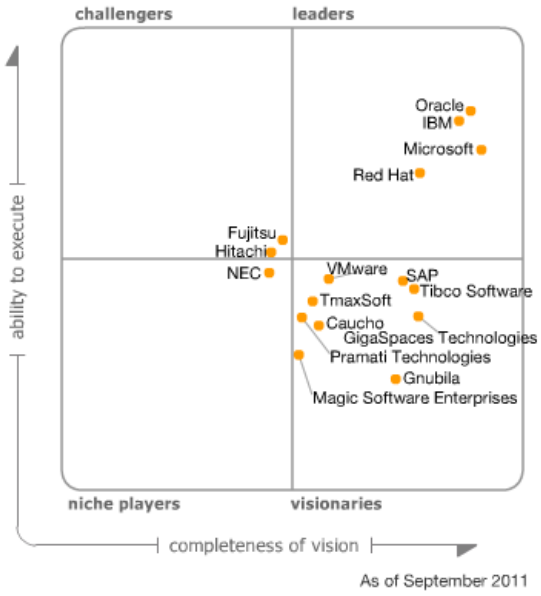


Figure 14

Key components of WebLogic include:

- Support for:
    - Web Sockets
    - OSGi
    - Spring
    - SAML
    - Web profile
    - Java EE 6

[7] Gartner Magic Quadrant for Enterprise Application Servers (September 2012). Source: Gartner, Magic Quadrant for Enterprise Application Servers (Massimo Pezzini, Yefim V. Natis, Kimihiko Iijima, Daniel Sholler, Raffaella Faveta – September 26, 2011). NOTE: This Magic Quadrant graphic was published by Gartner, Inc. as part of a larger research note and should be evaluated in the context of the entire report. The Gartner report is available here: http://www.gartner.com/technology/reprints.do?id=1-17GUO5Z&ct=110928&st=sb

- Advanced functionality including:
    - **JMS:** Choose from a host of persistence options, use distributed destinations, use multiple JDBC stores to concurrently process operations, and leverage Exalogic-only I/O-related optimizations
    - **Database connection pool management:** Set advanced rules for database connection pools that allow pools to automatically grow or contract based on real-time demand. Also leverage Active GridLink for RAC, which is described below
    - **Web services:** Quickly build out and manage RESTful, JAX-WS, JAX-RPC-based web services
    - **Work managers:** Auto-scale thread pools based on pre-defined rules
    - **Transaction management:** An advanced mature transaction manager with a multitude of configuration options
    - **Persistent JDBC TLOGs:** Persist transaction logs to a database and simplify disaster recovery
- Management tools such as:
    - WebLogic Server Administration Console
    - Oracle Enterprise Manager
    - WebLogic Scripting Tool (WLST)
    - JRockit Mission Control (if JRockit is used)
- Full vertical integration with the rest of Oracle's stack, including:
    - **Exalogic:** Natively leverage Exabus (Exalogic's InfiniBand-based I/O backplane) for inter-cluster communication, to connect to Oracle Traffic Director, and to connect to Oracle Exadata. There are also dozens of other WebLogic enhancements and optimizations that are only available on Exalogic
    - **Coherence:** Enable Coherence*Web (manages HTTP session data in a Coherence data grid) with the check of a box in the WebLogic Server Administration Console
    - **Enterprise Manager:** Fully manage WebLogic and other Oracle products (physical disks to Oracle middleware to Oracle applications) from a single user interface
    - **Tuxedo:** Leverage a complete integration to Tuxedo that optionally includes use of the Socket Direct Protocol on Exalogic
    - **Oracle RAC Database:** When making connections to an Oracle RAC database, WebLogic has a feature called Active GridLink for RAC which feeds Fast Application Notification (FAN) events in the database back to WebLogic using Oracle Notification Server (ONS). With WebLogic knowing what's happening in the

database, it's able to quickly route database connections to the appropriate RAC node based on availability, load, etc

Though WebLogic excels in performance and throughput, the real reason to choose a particular application server is its scalability – both in terms of technical scalability (input always being proportional to output) and its enterprise quality.

Caching

Caching is a key part of any large-scale eCommerce platform. Broadly, caching occurs at both the client-side and server-side.

Client-side Caching

Client-side caching, meaning caching that occurs on end-user's devices, is very important to ensure fast performance. Loading the home page of the top 10[8] U.S.-based eCommerce websites results in an average of 101 HTTP requests with a total weight of .95 MB. These numbers are only growing over time. Over the past year alone, the average page weight among the top 100 most visited websites in the U.S. has increased by a staggering 83%[9].

Let's look at an example. Sephora.com, an Oracle customer, has a home page that:

- Is 892 kilobytes in total size
- Requires 74 HTTP requests to load
- Takes 2.35 seconds to download and fully render

Of the 74 HTTP requests, only the first one actually goes back to Sephora's data center. Nothing occurs on the client-side until that HTTP request is returned. Oracle's platform for eCommerce is only responsible for servicing the HTTP request as quickly as possible. After that HTTP request is returned, the browser then parses the HTML to look for images, CSS files, JavaScript files, Flash files, etc to retrieve. It then retrieves those dozens or even hundreds of files in batches of roughly five HTTP requests at a time. Those static files are generally retrieved from a CDN.

[8] 2011 Internet Retailer Top 500 Guide

[9] http://httparchive.org

Graphically, the un-cached home page load looks like this:

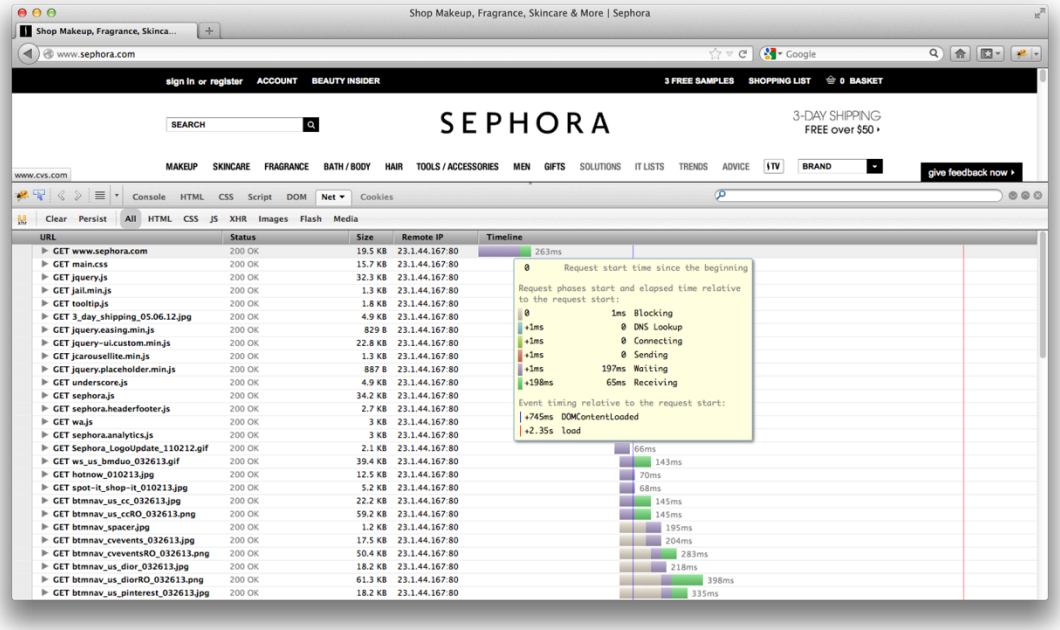


Figure 15

And at the end, you'll see:

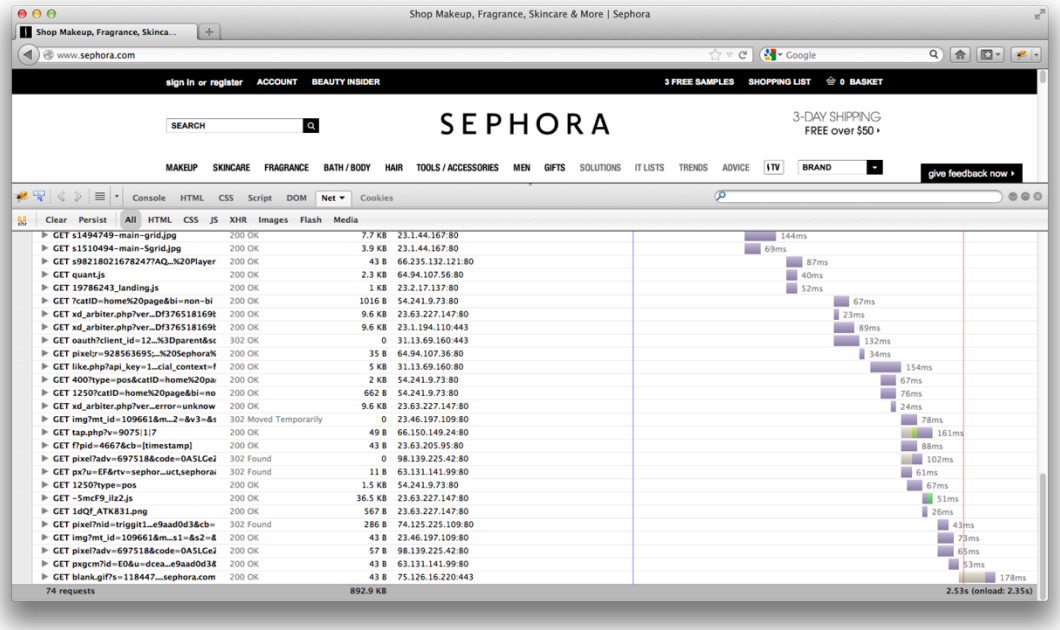


Figure 16

Notice how nothing happens until the initial HTTP request is returned. With appropriate caching (e.g. HTTP response headers configured so that clients cache the static media), subsequent page views tend to be much faster, with far fewer HTTP requests.

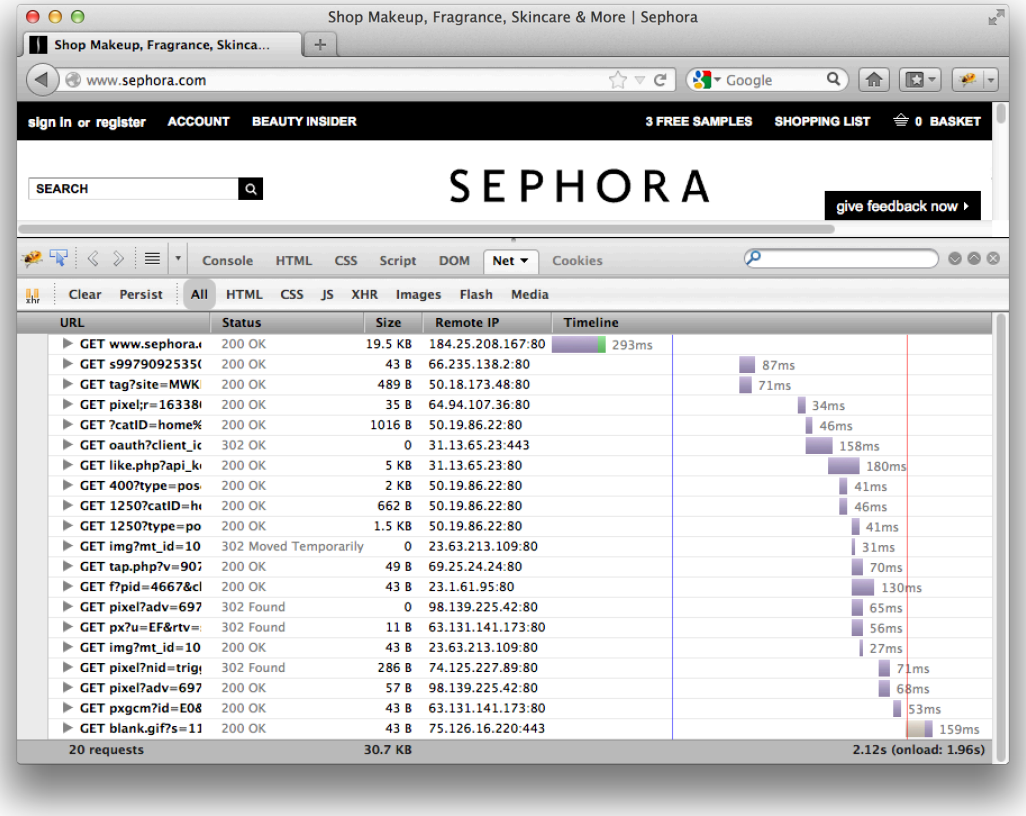


Figure 17

If a CDN is not used, Oracle HTTP Server makes it easy to configure cache headers appropriately in order to minimize the number of HTTP requests that are made with each page view.

Getting the first HTTP request of a page view to load is where Oracle's platform is most beneficial whereas a CDN is typically responsible for the other dozens or even hundreds of page views.

Server-side Caching

Since the initial HTTP request of a page view is blocking, it is important that it be as fast as possible. This is where Oracle's technology comes into play, as a CDN typically owns the delivery of everything else.

Caching is an important part of improving server-side response times. Caching can be enabled at the following tiers:

- **Load Balancing:** Oracle's load balancer, Oracle Traffic Director, offers the ability to cache static content as well as entire HTML responses so that HTTP requests don't even make it back to WebLogic
- **Web Serving:** Oracle's web server, Oracle HTTP Server, offers the ability to cache static content in order to offload requests to WebLogic
- **Application:** In-JVM caching (typically via a java.util.Map interface) has long been a part of application architecture. While beneficial for limited use cases, the approach doesn't scale because JVMs have a finite amount of heap (memory) available to them
- **Data Grid:** Oracle Coherence is the distributed in-memory data grid component of Oracle's Middleware suite that enables organizations to predictably scale mission-critical applications by providing fast and reliable access to frequently used data. By automatically and dynamically partitioning data in memory across multiple servers, Coherence enables continuous data availability and transactional integrity, even in the event of a server failure. As a shared infrastructure, Coherence combines data locality with local processing power to perform real-time data analysis, in-memory grid computations, and parallel transaction and event processing. Extensive use of caching results in less utilization of back-end systems, like databases, thus increasing horizontal scalability. By using Coherence*Web to manage HTTP sessions, memory in each application JVM is also saved, thus the increasing vertical scalability of each application instance

When looked at holistically the architecture is as follows:

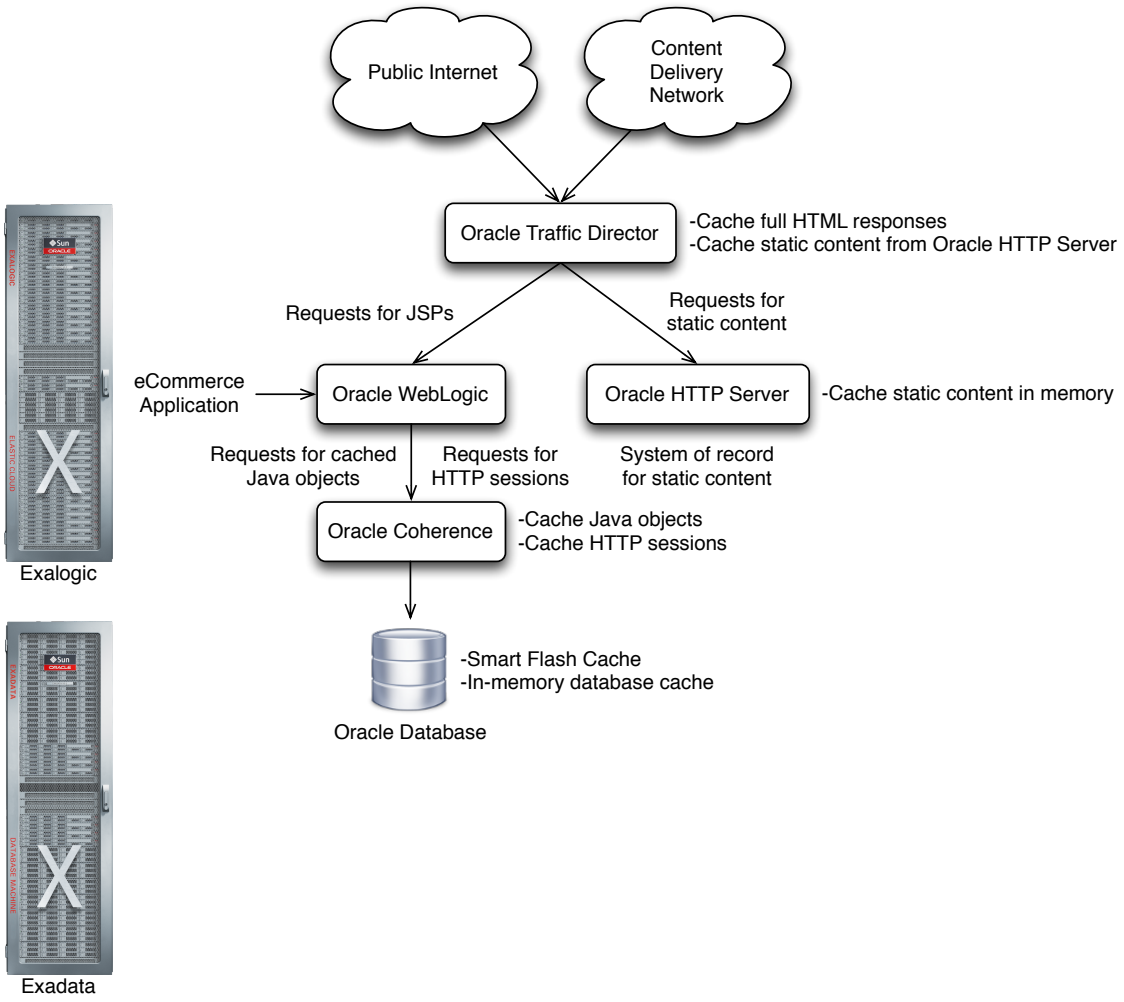


Figure 18

These tiers all work together to cache data intelligently. For example, Oracle Coherence natively leverages Exabus (InfiniBand networking technology and related protocols that are unique to Exalogic) through the use of InfiniBand's Remote Direct Memory Access (RDMA) feature. Leveraging an I/O backplane capable of 40 gigabits of throughput per second with 1.2 microsecond latency, Coherence can write over-the-wire into the memory of a separate physical server (compute node) within Exalogic. Coherence can also leverage 100 GB worth of solid-state drives per each 16-core Sun X3 compute node within Exalogic.

Application

While this white paper is oriented toward the platform underneath eCommerce applications, Oracle has a market-leading offering called Oracle Commerce.

ATG

Oracle ATG Web Commerce, or simply ATG, consists of a robust development framework, a large library of Java code that can be leveraged by developers, and a collection of management applications for business users. ATG ships headless, meaning UIs are custom-developed and implemented on top of the ATG framework. Various reference apps are shipped with the product but they're meant to be used as references and generally not as the starting point for a deployment.

ATG was built with three goals in mind:

1. Facilitate the creation of relevant shopping experiences for end-customers through advanced personalization
2. Give non-technical business users the ability to define and manage shopping experiences, data, and brand consistency across the different touch points
3. Extreme platform scalability

An over-arching goal of ATG's architecture is the ability to mix and match individual features between websites and channels into a single platform, in order to support true omni-channel retailing.

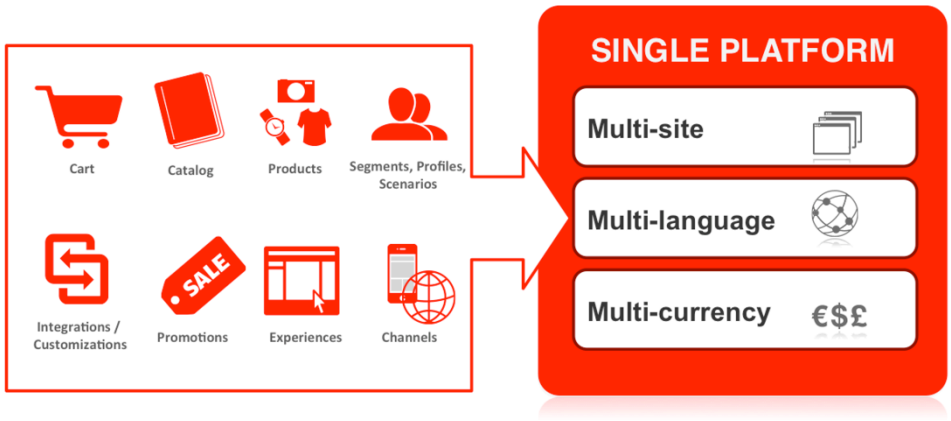


Figure 19

ATG is primarily used for B2C commerce through the web channel but it may also be used for B2B and C2C across channels ranging from mobile to embedded video game consoles. ATG uses a relational database as the system of record but many users of ATG use advanced analytics software and hardware from Oracle to supplement a relational database's capabilities.

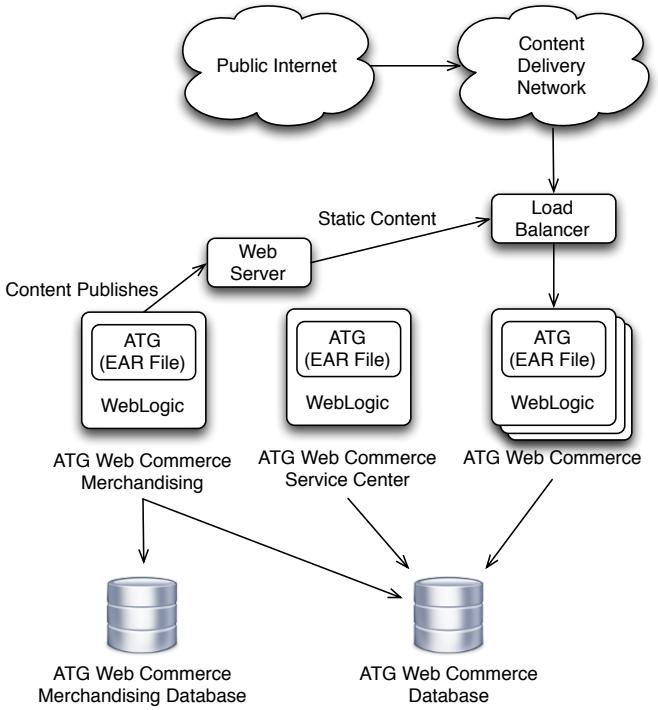
The major components of ATG include:

- Web Commerce: a series of APIs and Java libraries
    - Core eCommerce capabilities: Shopping cart, checkout, etc
    - Personalization: Targeting the right content to the right users at the right time
    - Pricing: B2C and B2B pricing, often based on personalization
    - Multi-site/micro-site: Quickly build new websites that target different audiences

    - Internationalization: Effectively reach different customers in different countries or locales

- Web Commerce Merchandising: An application complete with a friendly user interface
    - Business user-friendly interface for managing products, categories, prices, and other content
    - Ability to preview changes in-context prior to deployment
    - Ability to manage user segmentation and other personalization-related rules
- Commerce Service Center – an application complete with a friendly user interface
    - Interface for contact center agents to place orders, co-browse, respond to email questions

ATG's deployment architecture is typically as follows:



Endeca

Oracle Endeca, or simply Endeca, consists of an in-memory columnar database and a framework for loading and querying data. The columnar database is used for typical search queries (e.g. "red shirt"), faceted navigation queries (e.g. show all red shirts < $10), and for the placement of content on pages (e.g. which product should fill the hero image slot on the home page for this given user). Endeca is deployed as a series of standalone C/C++ executables accessible over web services. Individual Endeca search engines are entirely stateless, which simplifies the deployment architecture. In the course of generating a single page (e.g. the home page), Endeca may be queried dozens of time or more, always

through a load balancer. The richness of the user experience is directly correlated to how many queries go back to Endeca.

Endeca's deployment architecture is typically as follows:

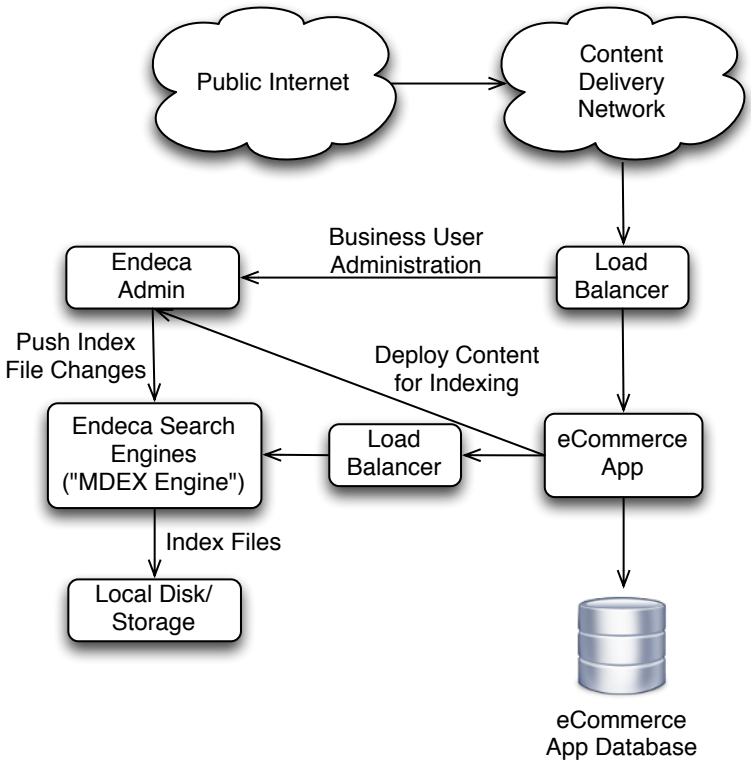


Figure 20

Endeca works exceptionally well on Exalogic, in particular because Exalogic offers load balancing between the eCommerce application and Endeca with virtually no overhead through Oracle Traffic Director running on Exalogic's Exabus I/O backplane.

Oracle Commerce

Oracle Commerce is a combination of Oracle ATG Web Commerce for eCommerce and Oracle Endeca for search and page management. The two products have each been substantially modified to work exceptionally well together, even though the two still may be used independently of each other.

The recommended deployment architecture of Oracle Commerce is:

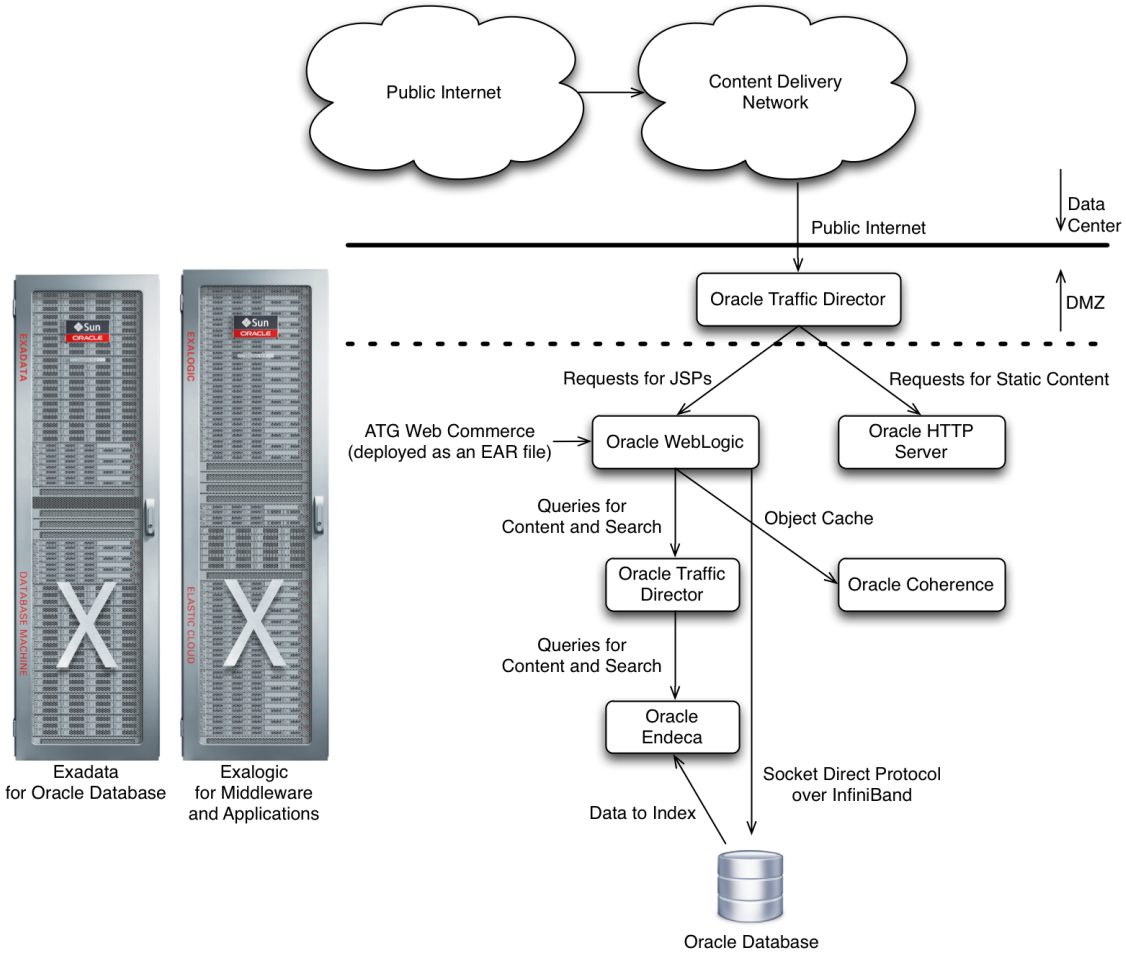


Figure 21

While the traditional deployment architectures that ATG and Endeca use have proven to be sufficient, Exalogic and Exadata bring benefits that are simply unachievable through other means.

Database

Large e-commerce applications, including those powered by Oracle Commerce, place unique demands on a database. ECommerce applications' use of a database is characterized by:

- High amounts of reading and writing (particularly from/to the same few tables)
- High numbers of transactions
- Short transaction lengths
- Queries for records almost exclusively by primary key(s)
- High frequency of queries with joins (though rarely more than three)
- Limited OLAP use

This section will focus on how to best scale your databases, while ensuring the highest possible availability.

Scaling Oracle Database

Oracle Database can be scaled vertically and, using Real Application Clusters (RAC) technology, horizontally. Often customers do both, achieving both high overall scalability and, through horizontal scaling, high availability.

Oracle RAC allows you to have many physical servers comprise a single logical database. Oracle RAC uses a "shared everything" architecture in which all data files, parameter files, and control files are all shared by all nodes in the cluster through use of a shared file system like a SAN. Redo logs and undo segments are individual to each node. Maintaining a consistent view of the data across all nodes requires an interconnect, which is a private network linking together nodes. As the number of nodes increases, the interconnect's finite bandwidth can become a bottleneck. The SAN used to house the data files can also limit throughput. Those realities limit the total number of nodes feasible for an Oracle RAC database. Exadata's InfiniBand-based interconnect substantially reduces these issues.

Even though the number of horizontal nodes is finite, each server can also be vertically scaled to increase throughput. Some of the high-end servers today have enough slots for 256 CPUs and terabytes of memory. The combined horizontal and vertical scaling allowed by Oracle RAC provides very high throughput.

Database Offload

Using replication strategies like Oracle Data Guard or Oracle GoldenGate, you can push read-only data from your highly available, writable database to a farm of separate non-clustered databases, known as slaves. Read-only data consists of data like categories, products, SKUs, and media. Data sources can then be configured to pull all of this read-only data from your slave databases, with different groups of application server instances reading from different slaves. Reporting applications could then be run off these slaves.

Not having to query your main clustered database for many of the repetitive read-only queries can significantly reduce its load. Since the slaves are just read-only copies, they do not need to be highly available or use any of the costly technologies used to achieve high availability and scalability in your main database. These slaves can fail without any data loss or impact to the end-customer.

Load Balancing

When you establish connections to your databases from your application server, you can choose to enable or disable load balancing. Enabling load balancing allows Oracle Database to dynamically load balance traffic between Oracle RAC nodes. Connections that are mostly read/write can significantly increase interconnect traffic. Excessive interconnect traffic can lead to database performance and throughput degradation and should therefore be avoided. For connections that are mostly read/write, disable load balancing but turn on failover. With HTTP session stickiness, any write operations to a database by a given user will happen to the same node.

Also specify the primary node, but be sure to alter what the primary node is. If all connections across your environment point to a given node as being the primary, and load balancing isn't enabled, you will run into problems. If your database consists of five nodes, configure 20% of nodes to connect to node 1, 20% to connect to node 2, etc. Make sure to change the order of the failover nodes as well.

Connections that are largely read-only, like to the catalog schemas, can use load balancing.

It is important to also properly define the JDBC URLs for each connection on each applications server instance. You cannot simply re-use the same JDBC URL across all instances.

Of course none of this is an issue if WebLogic's Active GridLink for RAC feature is used.

Partitioning Data

In addition to splitting up data by type, data of the same type can also be split up using Oracle Partitioning. Oracle partitioning would be necessary only under the most demanding of circumstances and when it is not possible to prune the data or prune the data fast enough. Good candidate tables may include email registration tables, order line item tables, or JMS-related tables.

Partitioning a table allows its data and indexes to be subdivided into smaller pieces, which can improve SQL query performance, maintenance, and management in certain circumstances. Queries that take minutes to execute can be shortened to seconds or even sub-second. Partitions can be stored in different tablespaces and created/modified/deleted at runtime, with no impact to performance or stability. As your dataset grows, you can re-balance your partitions at runtime. Data can be partitioned several different ways. For example, you might want to partition your dps_user table based on registration date (e.g. two months per partition), and your dms_msg table based on a hash. You can even configure multi and auxiliary tables (e.g. dps_user_address) to store their data in the same partition. This allows you to have all of a given customer's complete profile data in the same partition, which results in even better performance.

Another use of this technology would be partitioning your order table (dcspp_order) by date range, with the older partitions running on progressively less expensive and available hardware.

Operational Considerations

Problems in databases like locking, blocking, or deadlocking sessions are likely to cause serious problems for a large-scale eCommerce application in production. The threads in your application server that are executing those queries hold on to Java-level object locks, which are not released until the application receives a response from the database. To minimize the impact of this problem, set your transaction timeouts low in your application server's transaction manager and then quickly troubleshoot and fix the root cause. If possible, force your transaction manager to interrupt threads on transaction timeout so they don't get blocked. A good way of enforcing timeouts is to set the oracle.jdbc.ReadTimeout property of the JDBC driver and set the value to match the transaction timeout.

If a database session is blocked for some reason, Oracle's v$session table can tell you what physical server the session originated from, but that's it. With many instances per physical server, finding the instance that created the problematic session can be difficult. The solution is to set the session's

module on connection initialization. Most application servers can issue arbitrary SQL when establishing a connection to a database. With Oracle, the SQL would look like this:

```
begin dbms_application_info.set_module('instance=x, server=y,
application=z','');end;
```

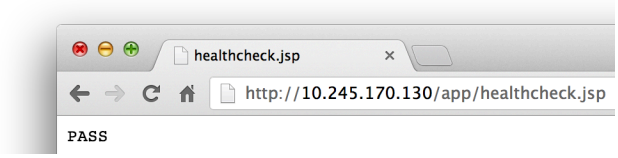
This calls a stored procedure to set the module for the session. The string passed to the set_module stored procedure shows up in v$session.module in AWR reports. When you have a session that's giving you problems, you can instantly see what specific application server instance the session originated from.

In addition to a transaction timeout with thread interruption on the application server, each database should implement a script to automatically kill transactions and individual DML statements that run for too long. The script should take into account the session's application (obtained from v$session.module), as different applications have varying transaction and DML execution lengths.

Monitoring

Monitoring must not be limited to whether pings are responded to but instead must be whether the application is actually functioning as expected. In other words, layers > 3 in the OSI model all must be monitored as opposed to just layer 3 and below. Comprehensive monitoring must be in place so that load balancers can make decisions on how to direct traffic. If there is a problem, traffic should be immediately redirected around it in order to meet desired SLAs and prevent a problem from becoming much larger and taking down an entire environment. Individual hardware and software components should be fully monitored, with alerts in place that trigger meaningful actions such as sending somebody an alert (a text message, an email message, etc) or taking a corrective action. Oracle Enterprise Manager supports all of these capabilities out of the box, including the monitoring of 3rd party products/services.

One of the most powerful features of Oracle Traffic Director and a CDN acting as a proxy is the ability to stop sending requests to an application server or even an entire data center if the application is not functioning properly but is still able to respond to TCP pings. To set up this monitoring properly, one or more JSPs should be written that exercises basic functionality like retrieving a product from the database and creating a new user profile. Upon execution, the JSP should return a string like "PASS" or "FAIL."



The load balancers can then be configured to look through the responses and to take corrective actions accordingly. For example, if two or more data centers are used, traffic can be shifted entirely to one in the event of an outage. There are numerous occasions where something will respond to a ping but still be incapable of servicing a customer's request.

In addition, all 3rd party integration points (those that are communicated with synchronously and asynchronously) must be monitored so that corrective action may be taken by code in the event of a failure or an event that will generate large volumes of traffic. For example, some customers disable 3rd party address verification on Black Friday and Cyber Monday because keeping it enabled would result in the failure of those 3rd party services. For synchronous integration points, knowing that a 3rd party service is down could allow code to asynchronously call the service as opposed to synchronously.

Exalogic and Exadata can "phone home" to Oracle and automatically open a support request in the event of a hardware issue. Proactively monitoring hardware ensures that Oracle is able to quickly send replacement hardware and engage Oracle Support.

Management

Management is just as important as monitoring because proper management can prevent problems from occurring. Doing management tasks manually is not practical or recommended because of the size of these environments. A large environment may contain hundreds or even thousands of CPU cores.

Management, provided by Oracle Enterprise Manager, consists of the following elements:

- **Configuration Management:** With Oracle Enterprise Manager, users log into a single user interface with individual accounts. Permissions are granted to each user or group by an administrator. All changes are made through the user interface. No more hand-editing XML files, which isn't scalable. Changes can be applied to the WebLogic domain, cluster or managed server with the click of a button. All changes are logged for auditing purposes

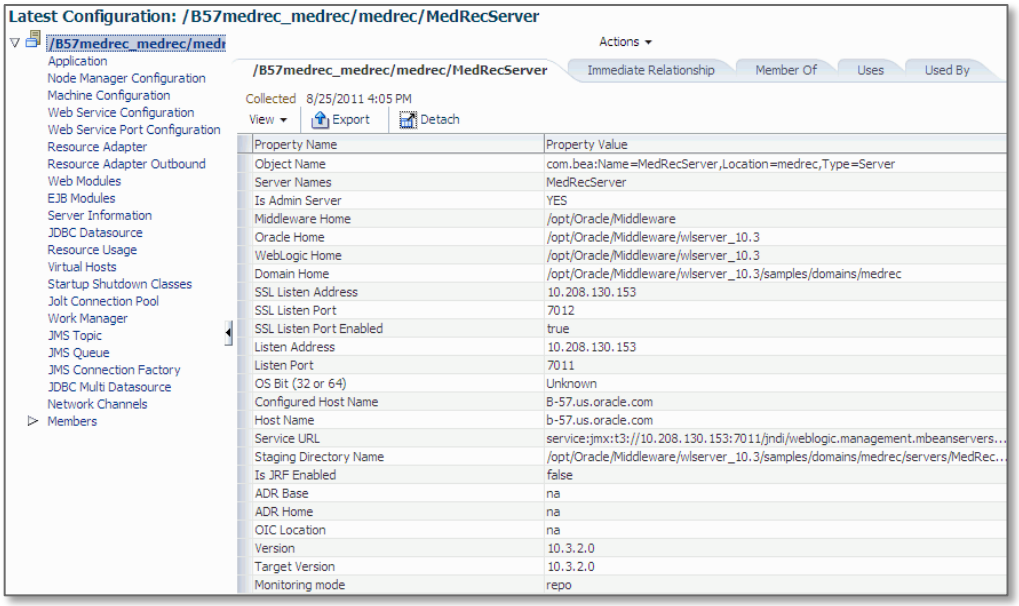


Figure 22

- **Patch Automation:** An integration with My Oracle Support allows for patch advisories to be automatically displayed so that critical patches can be applied quickly. Enterprise Manager provides a workbench that allows patches to be quickly and easily applied, without having to go to a command prompt
- **Provisioning:** Oracle Enterprise Manager allows for new capacity across the entire Oracle stack to be provisioned quickly and easily with the click of a few buttons. Capacity can even be dynamically added based on real-time resource needs

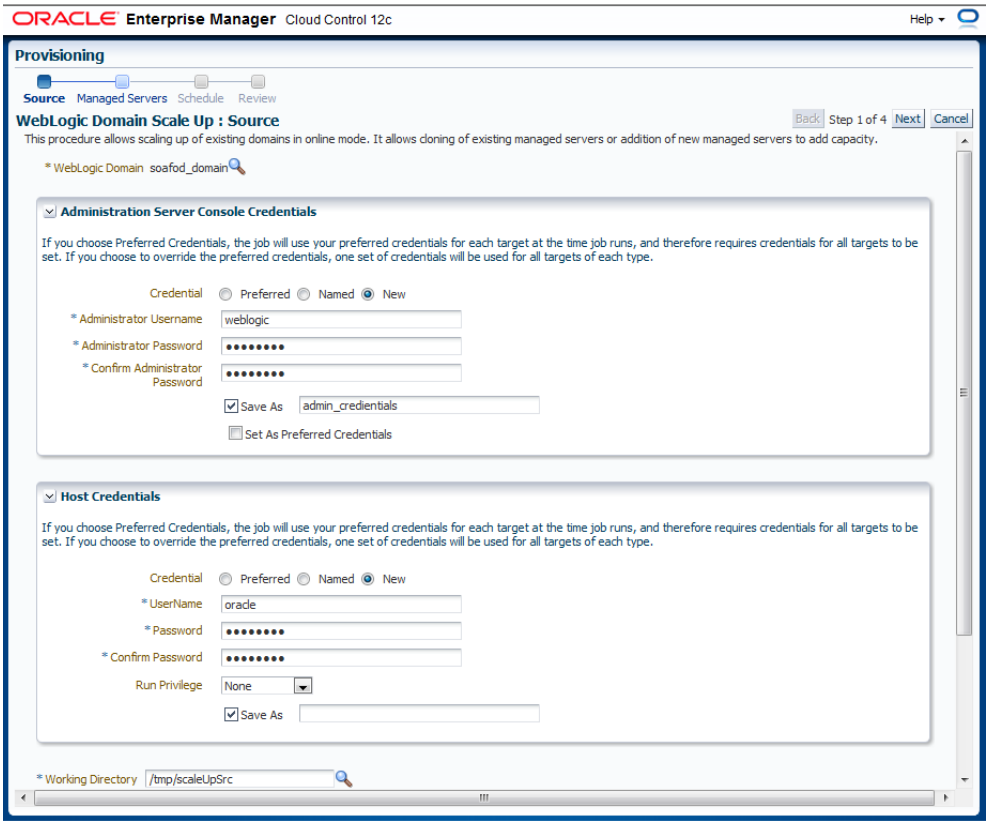


Figure 23

- **End-to-end Diagnostics:** Because Oracle's products are so tightly integrated, Oracle Enterprise Manager permits users to diagnose issues across all tiers from a single user interface

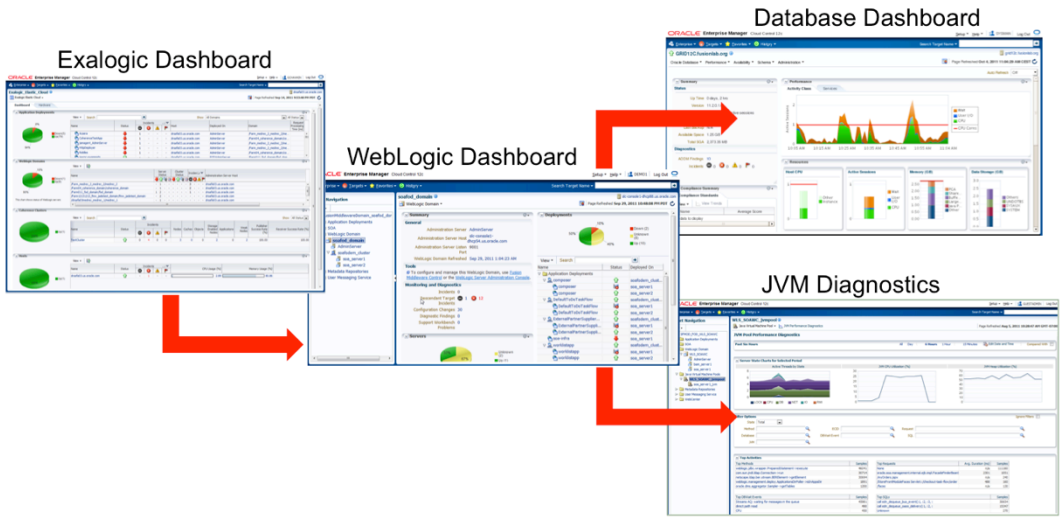


Figure 24

Supporting Architecture

With all of the new customer touch points, increasing expectations for high availability, and the increasing importance in eCommerce, designing platforms for scalability from the start is an imperative. This white paper outlined a series of simple, well-tested and pragmatic principles for designing large-scale eCommerce platforms.

Choosing Oracle as a partner ensures that you'll have the best people and technology at your side every step of the way. As eCommerce continues to grow and evolve, Oracle will continue to be at the leading edge of the space.

Building Large-Scale e-Commerce Platforms
With Oracle
April 3rd 2013
Author: Kelly Goetsch

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

SOFTWARE. HARDWARE. COMPLETE.